Building and Combining Matching Algorithms

Christophe Ringeissen

Building and Combining Matching Algorithms

Christophe Ringeissen[0000−0002−5937−6059]

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
Christophe.Ringeissen@loria.fr

Abstract. The concept of matching is ubiquitous in declarative programming and in automated reasoning. For instance, it is a key mechanism to run rule-based programs and to simplify clauses generated by theorem provers. A matching problem can be seen as a particular conjunction of equations where each equation has a ground side. We give an overview of techniques that can be applied to build and combine matching algorithms. First, we survey mutation-based techniques as a way to build a generic matching algorithm for a large class of equational theories. Second, combination techniques are introduced to get combined matching algorithms for disjoint unions of theories. Then we show how these combination algorithms can be extended to handle non-disjoint unions of theories sharing only constructors. These extensions are possible if an appropriate notion of normal form is computable.

Keywords: Matching · Unification · Combination of Theories · Syntactic Theories.

1 Introduction

Both unification and matching procedures play a central role in automated reasoning and in various declarative programming paradigms such as functional programming or (constraint) logic programming. In particular, unification is an essential engine in the execution of logic programs. In functional programming, functions are defined by pattern matching. In rule-based programming [20,18,39], matching is needed to apply a rule and so to perform a computation. In automated theorem proving [14,15,17,41,5,4], unification is applied to deduce new facts via expansion inferences, while matching is useful to simplify existing facts via contraction inferences. For the verification of security protocols, dedicated provers [28,37,16] handle protocols specified in a symbolic way. In these reasoning tools, the capabilities of an intruder are modeled using equational theories [1], and the reasoning is supported by decision procedures and solvers modulo equational theories, including equational unification and equational matching.

Unification and matching procedures aim at solving equations in term-generated structures [12,30]. A unification problem is a set of arbitrary equations between terms. A matching problem is a unification problem where each equation has a ground side, that is, a ground term possibly built over free constants.

Thus, a matching problem is a particular unification problem with free constants. In practice, syntactic unification, as well as syntactic matching, are particularly popular. In that singular case, the underlying equational theory is simply the empty theory and the well-known syntactic unification algorithm computes a most general solution when the input is solvable. More generally, we may consider equational unification and equational matching, where the problems are defined modulo an arbitrary equational theory, such as, for instance, one that defines a function symbol to be associative (A), commutative (C) or associative and commutative (AC). Equational unification and equational matching are undecidable in general. However, specialized techniques have been developed to solve both problems for particular classes of equational theories, many of high practical interest, including for example AC. The successful application of equational rewriting in rule-based programming languages [20,18,39] has demonstrated the interest of developing dedicated equational matching algorithms. Compared to unification, matching can be considered as a simpler problem. Hence A-matching is finitary, that is, the set of solutions of an A-matching problem is finite, whereas A-unification is infinitary. The decision problems for AC-matching and AC-unification are both NP-complete [32] even if for AC-matching the number of solutions is bounded by a single-exponential while a double-exponential [33] is needed to get a bound for AC-unification.

In this paper, we focus on the matching problem. We mainly consider the case of regular theories, that is, theories axiomatized by equalities such that both sides have the same set of variables. Matching in regular theories has a remarkable property: any solution of any matching problem is necessarily ground. This property eases the construction of matching algorithms. We survey two general techniques that allow us to design matching algorithms for a large class of (regular) theories.

First, we focus on mutation techniques that generalize the classical decomposition rule known from the syntactic case [30]. Using a more general mutation rule, it is possible to get a complete unification procedure for theories having the property of being syntactic [34,43]. Unfortunately, the resulting unification procedure only terminates for some particular classes of theories, such as shallow theories [21] or theories saturated by paramodulation [36]. However this unification procedure can be adapted to construct a matching procedure, which is actually terminating for the whole class of finite syntactic theories, as pointed out by Nipkow in the early 1990s [43]. Many permutative theories of practical interest for equational rewriting belong to that class, including A, C and AC [34,43]. For the class of finite syntactic theories, we present a rule-based matching algorithm along the lines of the classical rule-based syntactic unification algorithm.

Second, when a theory is defined as a union of theories, it is quite natural to consider methods that combine the matching algorithms available in the individual theories. In the early 1990s, a first combination method has been proposed by Nipkow for the matching problem in the union of disjoint regular theories [42]. Then the Baader-Schulz combination method has been a seminal contribution for the unification problem in the union of disjoint arbitrary

theories [11]. Compared to other combination methods previously developed by Schmidt-Schauss [49] and Boudet [19], the Baader-Schulz method permits us to solve both the unification problem and its related decision problem. Based on an approach à la Baader-Schulz, it is possible to develop new combination methods for the matching problem and its related decision problem, as shown in [46]. In this paper, we survey the existing combined matching methods, by showing how to reconstruct them thanks to the Baader-Schulz combination method and the underlying combination techniques. We also discuss their possible extensions to non-disjoint unions of theories, more precisely, theories sharing free constructors. We show that an approach à la Baader-Schulz can be applied to both the matching problem and the word problem in these non-disjoint unions. For the word problem, this leads to a method very similar to the one proposed by Baader and Tinelli [13] using an approach à la Nelson-Oppen [40].

The paper is organized as follows. Section 2 introduces the main technical concepts and notations. In Section 3, we present a mutation-based matching algorithm for a large class of syntactic theories. The combined unification problem is briefly discussed in Section 4, where we focus on the Baader-Schulz combination method. The combined word problem is detailed in Section 5 while Section 6 revisits the combined matching problem. In Section 7, we discuss the combined matching problem in the union of non-disjoint theories sharing free constructors. Eventually, Section 8 concludes with some final remarks about ongoing and future works.

2 Preliminaries

We use the standard notation of equational unification [12] and term rewriting systems [10]. A signature Σ is a set of function symbols with fixed arity. Given a signature Σ and a (countable) set of variables V, the set of Σ-terms over variables V is denoted by $T(\Sigma, V)$. The set of variables in a term t is denoted by $Var(t)$. A term is *linear* if all its variables occur only once. For any position p in a term t (including the root position ϵ), $t(p)$ is the symbol at position p, $t_{|p}$ is the subterm of t at position p, and $t[u]_p$ is the term t in which $t_{|p}$ is replaced by u. A substitution is an endomorphism of $T(\Sigma, V)$ with only finitely many variables not mapped to themselves. A substitution is denoted by $\sigma = \{x_1 \mapsto t_1, \ldots, x_m \mapsto t_m\}$, where the domain of σ is $Dom(\sigma) = \{x_1, \ldots, x_m\}$. Application of a substitution σ to t is written $t\sigma$.

A term t is *ground* if $Var(t) = \emptyset$. The set of ground Σ-terms is denoted by $T(\Sigma)$. When $\mathcal{C}$ denotes a finite set of constants not occurring in Σ, $\Sigma \cup \mathcal{C}$ is a signature defined as the union of Σ and $\mathcal{C}$, and $T(\Sigma \cup \mathcal{C})$ denotes the set of ground $(\Sigma \cup \mathcal{C})$-terms.

2.1 Equational Theories and Rewrite Systems

A Σ-axiom is a pair of Σ-terms, denoted by $l = r$. Variables in an axiom are implicitly universally quantified. Given a set E of Σ-axioms, the *equational theory*

$=_E$ presented by E is the closure of E under the laws of reflexivity, symmetry, transitivity, substitutivity and congruence (by a slight abuse of terminology, E is often called an equational theory). Equivalently, $=_E$ can be defined as the reflexive transitive closure $\leftrightarrow_E^*$ of an equational step $\leftrightarrow_E$ defined as follows: $s \leftrightarrow_E t$ if there exist a position p of s, $l = r$ (or $r = l$) in E, and substitution σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. An axiom $l = r$ is *regular* if $Var(l) = Var(r)$. An axiom $l = r$ is *permutative* if l and r have the same multiset of symbols (including function symbols and variables). An axiom $l = r$ is *linear* (resp., *collapse-free*) if l and r are linear (resp. non-variable terms). An equational theory is *regular* (resp., permutative/linear/collapse-free) if all its axioms are regular (resp., permutative/linear/collapse-free). An equational theory E is *finite* if for each term t, there are only finitely many terms s such that $t =_E s$. One can remark that a permutative theory is finite and a finite theory is regular and collapse-free. Well-known theories such as the associativity $A = \{x + (y + z) = (x + y) + z\}$, the commutativity $C = \{x + y = y + x\}$, and the associativity-commutativity $AC = A \cup C$ are permutative. Unification in permutative theories is undecidable in general [50].

A theory E is *syntactic* if it has a finite *resolvent presentation* S, defined as a finite set of axioms S such that each equality $t =_E u$ has an equational proof $t \leftrightarrow_S^* u$ with at most one step $\leftrightarrow_S$ applied at the root position. The theories A, C and AC are syntactic [43]. For C and AC, syntacticness can be shown as a consequence of the fact that any collapse-free theory is syntactic if it admits a unification algorithm [34].

A *term rewrite system* (TRS) is given by a set R of oriented Σ-axioms called rewrite rules and of the form $l \to r$ such that l, r are Σ-terms, l is not a variable and $Var(r) \subseteq Var(l)$. A term s *rewrites* to a term t w.r.t R, denoted by $s \to_R t$, if there exist a position p of s, $l \to r \in R$, and substitution σ such that $s_{|p} = l\sigma$ and $t = s[r\sigma]_p$. Given an equational theory E, $\longleftrightarrow_{R \cup E}$ denotes the symmetric relation $\leftarrow_R \cup \to_R \cup =_E$. A TRS R is Church-Rosser modulo E if $\longleftrightarrow_{R \cup E}^*$ is included in $\to_R^* \circ =_E \circ \leftarrow_R^*$. A reduction ordering $>$ is a well-founded ordering on terms closed under context and substitution. A reduction ordering $>$ is said to be E-compatible if $s > t$ implies $s' > t'$ for any terms s, t, s', t' such that $s' =_E s$ and $t' =_E t$. If $\to_R$ is included in an E-compatible reduction ordering, then there is no infinite sequence w.r.t $=_E \circ \to_R \circ =_E$ and according to [31] the following properties are equivalent:

1. R is Church-Rosser modulo E,
2. for any terms t, t', $t \longleftrightarrow_{R \cup E}^* t'$ if and only if $t \downarrow_R =_E \; t' \downarrow_R$, where $t \downarrow_R$ (resp., $t' \downarrow_R$) denotes any normal form of t (resp., t') w.r.t $\to_R$.

A substitution σ is *R-normalized* if, for every variable x in the domain of σ, $x\sigma$ is a normal form w.r.t $\to_R$.

2.2 Unification and Matching

From now on, we assume a signature Σ and a Σ-theory E such that Σ may include finitely many function symbols not occurring in the axioms of E. These

additional function symbols are said to be free in E. A Σ-equation is a pair of Σ-terms denoted by $s =^? t$. Variables in an equation are implicitly existentially quantified. When t is ground, an equation $s =^? t$ is called a match-equation, also denoted by $s \leq^? t$. An E-unification problem is a set of Σ-equations, $\Gamma = \{s_1 =^? t_1, \ldots, s_n =^? t_n\}$, or equivalently a conjunction of Σ-equations. We distinguish the following classes of E-unification problems Γ:

- if there is no free function symbol in Γ, then Γ is an elementary E-unification problem;
- if some free constants (resp., free symbols) occur in Γ, then Γ is an E-unification problem with free constants (resp., a general E-unification problem);
- if Γ is an E-unification problem with free constants (resp., a general E-unification problem) including only ground equations, Γ is an E-word problem (resp., a general E-word problem);
- if Γ is an E-unification problem with free constants (resp., a general E-unification problem) including only match-equations, Γ is an E-matching problem (resp., a general E-matching problem);
- If $\Gamma = \{x_1 =^? t_1, \ldots, x_n =^? t_n\}$ such that $x_1, \ldots, x_n$ are variables occurring only once in Γ, then Γ is called a solved form.

Consider any E-unification problem Γ. The set of variables in Γ is denoted by $Var(\Gamma)$. A solution to Γ, called an *E-unifier*, is a substitution σ such that $s_i\sigma =_E t_i\sigma$ for all $1 \leq i \leq n$. A substitution σ is *more general modulo E* than θ on a set of variables V, denoted as $\sigma \leq_E^V \theta$, if there is a substitution τ such that $x\sigma\tau =_E x\theta$ for all $x \in V$. A *Complete Set of E-Unifiers* of Γ, denoted by $CSU_E(\Gamma)$, is a set of substitutions such that each $\sigma \in CSU_E(\Gamma)$ is an E-unifier of Γ, and for each E-unifier θ of Γ there exists $\sigma \in CSU_E(\Gamma)$ such that $\sigma \leq_E^{Var(\Gamma)} \theta$. A class of E-unification problems is said to be finitary (resp., unitary) if any Γ in that class admits a $CSU_E(\Gamma)$ whose cardinality is finite (resp., at most 1). When E is an empty set of Σ-axioms, E is the empty Σ-theory denoted by $\emptyset$ where $\emptyset$-unification is unitary: a syntactic unification algorithm computes a $CSU_\emptyset(\Gamma)$ whose cardinality is at most 1 for any unification problem Γ.

Two signatures are disjoint if their respective sets of function symbols are disjoint. Two theories are disjoint if their respective signatures are disjoint. Given two disjoint signatures Σ_1 and Σ_2 and any $i = 1, 2$, Σ_i-terms (including the variables) and Σ_i-equations (including the equations between variables) are called *i-pure*. A term t is said to be *Σ_i-rooted* if its root symbol is in Σ_i. An *alien* subterm of a Σ_i-rooted term t is a Σ_j-rooted subterm s ($i \neq j$) such that all superterms of s are Σ_i-rooted. The position of an alien subterm of t is called an *alien* position of t. The set of alien positions of t is denoted by $APos(t)$. A term with at least one alien subterm is said to be *impure*.

3 Matching in Syntactic Theories

The interest of syntactic theories is to admit a mutation-based unification procedure that bears similarities with the rule-based unification algorithm known

for the empty theory [30]. In addition to the classical decomposition rule, additional mutation rules are needed, one for each equational axiom in a resolvent presentation of a syntactic theory. Unfortunately, this mutation-based unification procedure is not terminating for the class of syntactic theories. However, some important subclasses of syntactic theories actually admit a terminating mutation-based unification procedures, such as *shallow theories* [21], *forward-closed convergent theories* [24], and *equational theories saturated by paramodulation* [36]. When restricting to the matching problem, it is possible to get termination for a large class of theories of practical interest. Actually, Nipkow has shown that the class of finite syntactic theories admits a mutation-based matching algorithm presented as a Prolog-like program in [43]. We give in Fig. 1 a rule-based presentation of this mutation-based matching algorithm. An implementation for the AC case of this rule-based algorithm has been studied in the $UNIF_{AC}$ system developed in Nancy in the early 1990s [2,3]. As shown in [35], this AC-matching algorithm can be easily prototyped using a rule-based programming environment.

Theorem 1. *Consider the* MFS *inference system given in Fig. 1 where* Mutate *is assumed to be applied in a non-deterministic way in addition to* Dec. *The* MFS *inference system provides a mutation-based matching algorithm for any finite syntactic theory admitting a resolvent presentation S.*

Alternatively, there exists a brute force method to get a matching algorithm for finite theories via a reduction to syntactic matching: the finite set of substitutions $\{\sigma \in CSU_\emptyset(s =^? t') \mid t' =_E t\}$ is a $CSU_E(s \leq^? t)$. Compared to this brute force method, the interest of MFS is to show that a slight adaptation of the classical syntactic matching algorithm, i.e., the addition of a single rule, is sufficient to get a matching algorithm for the class of finite syntactic theories. One can notice that MFS can be turned into a decision procedure for the word problem. Moreover, the class of finite syntactic theories being closed by disjoint union [43], MFS can be applied for any union of disjoint finite syntactic theories. To consider more general unions of disjoint theories, we need to rely on combination methods discussed in the rest of the paper.

4 Unification in Unions of Disjoint Theories

There exist several combination methods for the unification problem, in which we find different forms of unification: elementary unification, unification with free constants and unification with free function symbols (also called general unification). Each of these combination methods corresponds to a given class of theories: regular collapse-free theories, regular theories and arbitrary theories. We briefly recall the modularity results that can be derived from these combination methods.

Theorem 2. *The following modularity results are consequences of existing combination methods:*

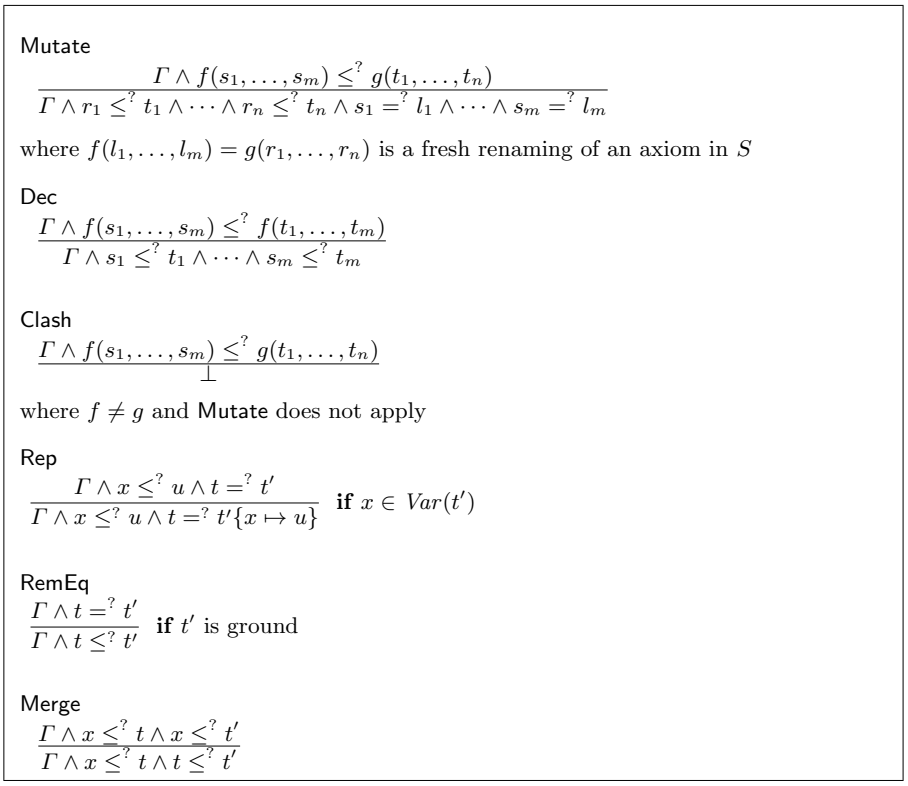

Fig. 1. MFS matching algorithm for finite syntactic theories

1. *The class of* **regular collapse-free theories** *admitting an* **elementary unification algorithm** *is closed under disjoint union [51,53].*
2. *The class of* **regular theories** *admitting a* **unification algorithm with free constants** *is closed under disjoint union.*
3. *The class of* **equational theories** *admitting a* **general unification algorithm** *is closed under disjoint union [49,11].*
4. *The class of* **equational theories** *admitting a* **general unification decision procedure** *is closed under disjoint union [11].*

We briefly outline the principles of a combination method for the unification problem in a union of two disjoint theories. First, the input problem is separated into two pure problems. Then, solutions of the pure problems must be carefully combined in order to construct solutions for the input problem. Two cases may appear.

Conflict of theories: The same variable can be instantiated simultaneously in both theories. To solve this conflict, the solution is to select the theory in which the variable is instantiated, meaning that it will be considered as a free constant in the other theory. This transformation of variables into free

constants requires an identification of variables to take care of the fact that two variables equally instantiated in one theory must be considered as the same free constant in the other theory. Then, applying unification algorithms with free constants is sufficient to avoid all these conflicts of theories.

Compound cycle: the conjunction of two pure solved forms can be a compound cycle such as $x_1 = t_1[x_2] \wedge x_2 = t_2[x_1]$ where t_i is i-pure for $i = 1, 2$.

To tackle both the conflicts of theories and the compound cycles, the Baader-Schulz combination method [11] considers a general form of unification called unification with linear constant restriction. It has been shown in [11] that unification with linear constant restriction and general unification are two equivalent notions, leading to the modularity result of the general unification problem given in Theorem 2. In the combination method proposed by Schmidt-Schauss [49], each pure problem is solved in its theory thanks to a unification algorithm with free constants together with a constant elimination algorithm. Actually, constant elimination is useful to break compound cycles [19]. The Schmidt-Schauss method combines unification algorithms while the Baader-Schulz method is also able to combine unification decision procedures. A major application of the Baader-Schulz combination method is to provide a way to show the decidability of general A-unification. The combination techniques developed by Baader and Schulz allow us to reconstruct the combination methods known for regular collapse-free theories and for regular theories:

- for collapse-free theories, a conflict of theories has no solutions,
- for regular theories, a compound cycle has no solutions.

Hence, for regular collapse-free theories, both the conflicts of theories and the compound cycles have no solutions. In the following, we show how to apply the Baader-Schulz combination method and the underlying techniques to build combination methods for two particular unification problems with free constants: the word problem and the matching problem.

5 The Word Problem in Unions of Disjoint Theories

In this section we consider unification problems with free constants where all equations are ground. In other words, we are interested in checking the equality of terms modulo an equational theory, that is, deciding the word problem. The development of a disjoint combination method for the word problem has been considered in [51,49,42] as a first step before investigating more general combination problems. Actually, it was already successfully addressed in [45].

The Baader-Schulz combination method can be applied to reconstruct a combination method dedicated to the word problem, where the word problem is viewed as a particular unification problem with (free) constants for which the theory selection can be simplified and no linear constant restriction is needed. To get a deterministic theory selection, it is useful to normalize the layers related to theories occurring in an impure term.

Definition 1. *An impure term t is in* layer-reduced form *if its alien subterms are in layer-reduced form and if t is not equal to one of its alien subterms. A pure term is in* layer-reduced form *if it is not equal to one of its variables or free constants.*

Example 1. Let $E_1 = \{x + 0 = x, 0 + x = x\}$ and $E_2 = \{g(x, x) = x\}$. The term $g(a, g(a+0, 0+a))$ is not in layer-reduced form but $g(a, g(a+0, 0+a)) =_{E_1 \cup E_2} a$ where a is in layer-reduced form. The term $g(a, b) + g(a, a + 0)$ is not in layer-reduced form but $g(a, b) + g(a, a + 0) =_{E_1 \cup E_2} g(a, b) + a$ where $g(a, b) + a$ is in layer-reduced form.

1. Purify
   Apply as long as possible the following rule:

   VA
   $$\frac{\exists \bar{v} : \Gamma \wedge s =^? t}{\exists y, \bar{v} : s =^? t[y]_p \wedge y =^? t_{|p}} \quad \textbf{if } \{s\} \cap \bar{v} = \emptyset, p \in APos(t), y \text{ is a fresh variable}$$

2. Identify
   Apply as long as possible the following rule:

   $$\frac{\exists y, y', \bar{v} : \Gamma \wedge y =^? u \wedge y' =^? u'}{\exists y, \bar{v} : \Gamma\{y' \mapsto y\} \wedge y =^? u} \quad \textbf{if } u =_{E_1 \cup E_2} u'$$

Fig. 2. Abstract algorithm

Lemma 1. *Let s and t be two terms in layer-reduced form. If s and t are free constants, $s =_{E_1 \cup E_2} t$ iff $s = t$. If s is Σ_i-rooted and t is not Σ_i-rooted, then $s \neq_{E_1 \cup E_2} t$. If s and t are both Σ_i-rooted, the* Abstract *algorithm (cf. Fig. 2) applied to $s =^? t$ returns a set of equations including only one i-pure equation $s_i =^? t_i$ between Σ_i-rooted terms such that $s =_{E_1 \cup E_2} t$ iff $s_i =_{E_i} t_i$.*

Lemma 1 provides a recursive combination method for the word problem. A non-recursive version would purify all the alien subterms and would be followed by a variable identification phase to identify variables denoting pure terms that are equal in the related component theory.

Lemma 1 assumes that the input terms are in layer-reduced form. If we have decision procedures for the word problem in the individual theories of the considered union, then the computation of an equivalent term in layer-reduced form is effective by using a bottom-up process which consists in repeatedly checking whether a pure term is equal to one of its variable or free constant. Let us call LRF the algorithm obtained from Abstract (cf. Fig. 2) by adding the steps depicted in Fig. 3 after Identify.

3. Collapse
Apply the following rule:

$$\frac{\exists \bar{v} : \Gamma \wedge x =^? t}{\exists \bar{v} : \Gamma \wedge x =^? c} \textbf{ if } \begin{cases} \{x\} \cap \bar{v} = \emptyset \\ t \text{ is a non-variable } i\text{-pure term} \\ c \text{ is a free constant or a variable in } t \text{ such that } t =_{E_i} c \end{cases}$$

4. Eliminate
Apply as long as possible the following rule:

$$\frac{\exists y, \bar{v} : \Gamma \wedge y =^? u}{\exists \bar{v} : \Gamma\{y \mapsto u\}}$$

Fig. 3. Collapsing for the layer-reduced form computation

Lemma 2. *Let t be any Σ_i-rooted term. Assume all the aliens subterms of t are in layer-reduced form. Given a variable x not occurring in t, the* LRF *algorithm applied to $x =^? t$ returns an equation $x =^? u$ such that $u =_{E_1 \cup E_2} t$ and u is in layer-reduced form.*

By Lemma 1 and Lemma 2 we get the following modularity result:

Theorem 3. *The class of* **equational theories** *admitting a* **decision procedure for the word problem** *is closed by disjoint union.*

6 Matching in Unions of Disjoint Theories

We now study the design of (disjoint) combination methods for the matching problem. To get a dedicated combination method, it is not sufficient to plug matching algorithms into a combination method initially designed for the unification problem. Indeed, the purification phase does not preserve the property of being a matching problem, and so we would have to solve pure equational problems that are not just matching problems. We focus on two simple cases where it is possible to generate equational problems that can be solved thanks to matching algorithms.

6.1 Regular Collapse-Free Theories

In the particular case of regular collapse-free theories, the purification phase can be adapted to introduce only match-equations instead of solved equations. Consider an $E_1 \cup E_2$-matching problem $\{s \leq^? t\}$ where s is impure. Suppose σ is a substitution such that $s\sigma =_{E_1 \cup E_2} t$. When E_1 and E_2 are regular collapse-free, $s\sigma$ and t are rooted in the same theory and any alien subterm of $s\sigma$ is $E_1 \cup E_2$-equal to some alien subterm of t which is necessarily ground. Thus, any alien subterm of s can be unified with some ground alien subterm of t. This leads to a

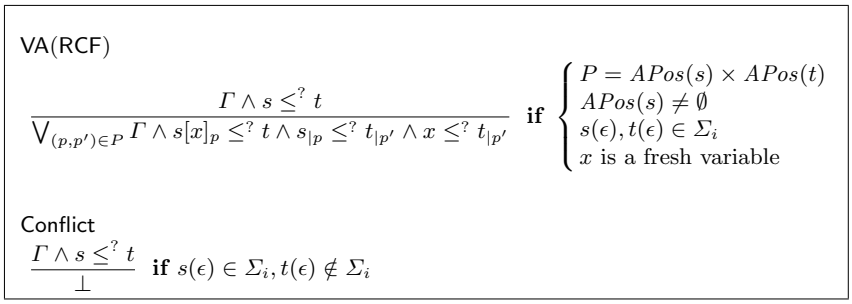

Fig. 4. Purification for regular collapse-free theories

particular purification phase (cf. Fig. 4) producing *left-pure* matching problems that can be handled by the Baader-Schulz combination method. Consequently, it is sufficient to use matching decision procedures.

Theorem 4 ([46]). *The class of* **regular collapse-free theories** *admitting a* **matching decision procedure** *is closed under disjoint union.*

6.2 Regular Theories

Following the approach initiated by Nipkow [42], we present a deterministic combination method described by the inference system given in Fig. 5. Here, we do not care about introducing a pending equation $x =^? s$. In regular theories, this variable x occurring elsewhere in a match-equation will be eventually unified with a ground term. Indeed, solving a match-equation including x generates a conjunction of solved match-equations, in particular a match-equation of the form $x \leq^? t$. Thus, the pending equation $x =^? s$ can be turned into the match-equation $s \leq^? t$.

Theorem 5 ([42]). *The class of* **regular theories** *admitting a* **matching algorithm** *is closed under disjoint union.*

6.3 Arbitrary Theories

The combination of regular theories with linear ones is problematic as shown in the following example borrowed from [42].

Example 2. Consider the two theories $E_1 = \{f(f(x)) = a\}$ and $E_2 = \{g(x, x) = x\} \cup DA$ where

$$DA = \begin{cases} x + (y + z) = (x + y) + z \\ x * (y + z) \ = x * y + x * z \\ (x + y) * z \ = x * z + y * z \end{cases}$$

The theory E_1 is a linear theory where the unification with free constants is decidable. The theory E_2 is a union of two disjoint regular theories, each of

LeftVA

$$\frac{\Gamma \wedge s \leq^? t}{\Gamma \wedge s[x]_p \leq^? t \wedge x =^? s_{|p}} \quad \textbf{if } p \in APos(s), x \text{ is a fresh variable}$$

Merge

$$\frac{\Gamma \wedge x \leq^? t \wedge x =^? s}{\Gamma \wedge x \leq^? t \wedge s \leq^? t}$$

Match

$$\frac{\Gamma \wedge s \leq^? t}{\bigvee_{\sigma \in CSU_{E_i}(s \leq^? t)} (\Gamma \wedge \bigwedge_{x \in Dom(\sigma)} x \leq^? x\sigma)} \quad \textbf{if } s \text{ is a non-variable } i\text{-pure term}$$

Delete

$$\frac{\Gamma \wedge x \leq^? t \wedge x \leq^? t'}{\Gamma \wedge x \leq^? t} \quad \textbf{if } t =_E t'$$

Fail

$$\frac{\Gamma \wedge x \leq^? t \wedge x \leq^? t'}{\bot} \quad \textbf{if } t \neq_E t'$$

For any $s \leq^? t$ in the above rules, t is supposed to be in layer-reduced form.

Fig. 5. Matching for the union of regular theories

them admitting a matching algorithm. Thus, the combined method presented in Section 6.2 can be applied to get an E_2-matching algorithm. However $E_1 \cup E_2$-matching is undecidable since for any terms s and t built over the signature of DA, the $E_1 \cup E_2$-matching problem $\{f(g(f(s), f(t))) \leq^? a\}$ has a solution iff the DA-unification problem $\{s =^? t\}$ has a solution. Since DA-unification is undecidable, $E_1 \cup E_2$-matching is undecidable while E_1-matching and E_2-matching are both decidable.

In [46], we give a combination method à la Baader-Schulz for the matching problem and its related decision problem. This method is complete for a large class of problems, like matching problems in *partially linear* theories, which are an extension of linear theories including regular collapse-free theories. In the class of partially linear theories, applying matching algorithms is sufficient, the linear constant restriction being superfluous even for the combination of matching decision procedures.

6.4 Matching versus General Matching

The combination method for the matching problem in regular theories E can be applied to construct a general E-matching algorithm from an E-matching algorithm. This leads to a natural question: is there an equational theory for which matching is decidable while general matching is not? A positive answer

to this question is given in [47], by considering a (many-sorted) theory that includes DA (cf. Example 2). This result shows that a combination method for the matching problem cannot exist for arbitrary theories. In the same vein, a similar question arises when comparing unification with free constants and general unification: is there an equational theory for which unification with free constants is decidable while general unification is not? A positive answer to this question is given in [44].

7 Matching in Unions of Non-Disjoint Theories

We discuss the problem of designing combination methods for unions of theories sharing constructors [23], by focusing on the word problem and the matching problem. To formalize the notion of constructor, it is convenient to rely on a rewrite system, where a constructor is simply a function symbol not occurring in root positions of left-hand sides. However, not every equational theory can be equivalently presented by a rewrite system. Fortunately, it is always possible to rely on a rewrite system that could be obtained by unfailing completion [11]. Alternatively, this rewrite system and the related constructors can be defined with respect to a reduction ordering over a combined signature used to orient combined ground instances of pure valid equalities. We consider below equational rewrite systems to cope with constructors modulo an equational theory E_0. In the case of absolutely free constructors considered in [23], the theory E_0 is empty.

Definition 2. *Let E_i be a Σ_i-theory for $i = 0, 1, 2$ and $\Sigma = \Sigma_1 \cup \Sigma_2$. The theory $E_1 \cup E_2$ is said to be a* combination of theories sharing constructors modulo E_0 *if $\Sigma_0 = \Sigma_1 \cap \Sigma_2$ and for any arbitrary finite set of variables V viewed as free constants, there exists an E_0-compatible reduction ordering $>$ on the set of ground $(\Sigma \cup V)$-terms $T(\Sigma \cup V)$ satisfying the following two properties for the set R_i $(i = 1, 2)$ of rewrite rules $l\psi \to r\psi$ such that $l\psi > r\psi$; l, r are Σ_i-terms, $l =_{E_i} r$, l is $(\Sigma_i \setminus \Sigma_0)$-rooted; $l\psi$ and $r\psi$ are ground $(\Sigma \cup V)$-terms thanks to a (grounding) substitution ψ:*

1. *$\longleftrightarrow^*_{R_i \cup E_0}$ coincides with $=_{E_i}$ on $T(\Sigma \cup V)$,*
2. *R_i is Church-Rosser modulo E_0 on $T(\Sigma \cup V)$.*

Example 3. To satisfy Definition 2, it is sufficient to consider a Σ_0-theory E_0 plus two finite TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$ over respectively Σ_1 and Σ_2 such that

- there is no Σ_0-symbol occurring at the root position of any left-hand side of $\mathcal{R}_1 \cup \mathcal{R}_2$,
- $\mathcal{R}_1 \cup \mathcal{R}_2$ is included in an E_0-compatible reduction ordering,
- $\mathcal{R}_1$ and $\mathcal{R}_2$ are both Church-Rosser modulo E_0.

Then E_0, $E_1 = \mathcal{R}_1 \cup E_0$ and $E_2 = \mathcal{R}_2 \cup E_0$ fulfill Definition 2 using the ordering $>$ provided by the transitive closure of $=_{E_0} \circ \to_{\mathcal{R}_1 \cup \mathcal{R}_2} \circ =_{E_0}$.

From now on, we assume that $\Sigma = \Sigma_1 \cup \Sigma_2$ and $E = E_1 \cup E_2$ is a combination of theories sharing constructors modulo E_0, where R_1 and R_2 denote the TRSs introduced in Definition 2. According to this definition, for any $f \in \Sigma_0$, and any terms $t_1, \ldots, t_m$ in $T(\Sigma \cup V)$, $f(t_1, \ldots, t_m) \downarrow_{R_i} =_{E_0} f(t_1 \downarrow_{R_i}, \ldots, t_m \downarrow_{R_i})$. The combined TRS defined by $R = R_1 \cup R_2$ satisfies the following properties: the rewrite relation $(=_{E_0} \circ \to_R \circ =_{E_0})$ is terminating, $(\longleftrightarrow_R \cup =_{E_0})^*$ coincides with $=_E$ on $T(\Sigma \cup V)$ and R is Church-Rosser modulo E_0 on $T(\Sigma \cup V)$. Thus, for any terms $s, t \in T(\Sigma \cup V)$, $s =_E t$ iff $s \downarrow_R =_{E_0} t \downarrow_R$, and for any $f \in \Sigma_0$, and any terms $t_1, \ldots, t_m$ in $T(\Sigma \cup V)$, $f(t_1, \ldots, t_m) \downarrow_R =_{E_0} f(t_1 \downarrow_R, \ldots, t_m \downarrow_R)$. R-normal forms are useful to define the notion of variable abstraction in a way similar to [11].

Definition 3 (Variable Abstraction). *Let W be a set of variables such that V and W are disjoint. Let $\pi : \{t \downarrow_R \mid t \in T(\Sigma \cup V), t \downarrow_R \notin V\} \longrightarrow W$ be a bijection called a* variable abstraction with range W. *For $i = 1, 2$, the* i-abstraction *of t is denoted by t^{π_i} and defined as follows:*

- *If $t \in V$, then $t^{\pi_i} = t$.*
- *If t is a Σ_i-rooted term $f(t_1, \ldots, t_n)$, then $t^{\pi_i} = f(t_1^{\pi_i}, \ldots, t_n^{\pi_i})$.*
- *Otherwise, if $t \downarrow_R \notin V$ then $t^{\pi_i} = \pi(t \downarrow_R)$ else $t^{\pi_i} = t \downarrow_R$.*

The notion of variable abstraction is instrumental to state technical lemmas showing that unification and matching procedures known in component theories can be reused without loss of completeness in the combination of theories sharing constructors modulo E_0.

Lemma 3 (Unification). *Consider any $i = 1, 2$, any i-pure terms s and t, and any R-normalized substitution σ. We have that $s\sigma =_E t\sigma$ iff $s\sigma^{\pi_i} =_{E_i} t\sigma^{\pi_i}$.*

In general, R is infinite and so it may be difficult to assume the computability of R-normal forms. In practice, we can rely on a notion of layer-reduced form, just like in the disjoint case. In this non-disjoint setting, a term t is said to be in *layer-reduced* normal form if $t^{\pi_i} =_{E_i} (t \downarrow_R)^{\pi_i}$ for any $i = 1, 2$. Let us assume that for any term, it is possible to compute an E-equal term in layer-reduced form.

Lemma 4 (Word problem). *Consider any $i = 1, 2$ and any terms s and t in layer-reduced form. We have that $s =_E t$ iff $s^{\pi_i} =_{E_i} t^{\pi_i}$.*

Notice that Abstract (cf. Fig. 2) applied to $s =^? t$ computes an i-pure equation which is a renaming of $s^{\pi_i} =^? t^{\pi_i}$ when s and t are Σ_i-rooted terms in layer-reduced form.

Lemma 5 (Matching). *Consider any $i = 1, 2$, any i-pure term s, any term t in layer-reduced form and any R-normalized substitution σ. We have that $s\sigma =_E t$ iff $s\sigma^{\pi_i} =_{E_i} t^{\pi_i}$.*

By Lemma 5 and assuming the computability of layer-reduced forms, the combination methods developed for the matching problem in the union of disjoint theories (cf. Section 6.1 and Section 6.2) can be reused to obtain:

- a combination method for matching decision procedures, deciding in a modular way the matching problem in the combination of regular collapse-free theories sharing constructors (modulo E_0);
- a combination method for matching algorithms, solving in a modular way the matching problem in the combination of regular theories sharing constructors (modulo E_0).

The results presented in this section rely on the use of R-normal forms. The word problem in unions of theories sharing non-absolutely free constructors has been successfully studied in [13], by introducing a computable notion of G-normal forms where G is a particular set of generators called Σ_0-base. In the above setting, a Σ_0-base G corresponds to the set of R-normalized terms that are not Σ_0-rooted. We have not discussed how to compute layer-reduced forms. A possibility is to build them by using normal forms that can computed in component theories, like in [13] for the computation of G-normal forms. The case of absolutely free constructors has been initiated in [23], with some preliminary results for the word problem and the matching problem. Then, a particular form of non-absolutely free constructors has been investigated for a class of theories sharing "inner" constructors, by focusing on the matching problem [48].

More recently, a form of hierarchical combination has been considered in [26]. In that case, the combined theory is given by a term rewrite system R_1 together with an equational Σ_2-theory E_2 such that Σ_2-symbols can occur only below the root positions of right-hand sides of R_1. Thus, under appropriate assumptions on R_1 and E_2, it is possible to design a combination method leading to a $R_1 \cup E_2$-matching algorithm [26]. This procedure uses the combination rules of Fig. 5, the decomposition rules of Fig. 1 for R_1, and applies an E_2-matching algorithm.

8 Conclusion

In this paper, we survey general techniques to build equational matching algorithms for a large class of (combined) theories of practical interest, e.g., in rule-based programming [20,18,39]. Furthermore, we show that the non-disjoint combination of matching procedures can be envisioned when the combined theory admits a computable notion of normal form. The non-disjoint combination of unification procedures remains a challenging problem. There are preliminary results for particular classes of theories such as shallow theories [26] and forward-closed theories [24]. For these particular classes of theories, a mutation-based approach [21] or a variant-based approach [22,29,38] can be successfully applied to solve the (combined) unification problem, but we believe it is always interesting to point out a combination-based alternative when the background theory is a union of "separable" theories. As shown here with the matching problem, some particular decision problems can admit non-disjoint combination methods. In that direction, non-disjoint combination methods have been developed in [27] for two decision problems related to (context) unification and of practical interest in the analysis of security protocols, namely the deduction problem and the indistinguishability problem [1].

Acknowledgments. I am grateful to the reviewers for their insightful remarks.

I would like to thank Hélène Kirchner and Claude Kirchner who gave me the opportunity to start doing research in equational theorem proving, and all my co-authors involved in joint works in connection to this survey:

- Unification in constructor-sharing equational theories [23]: Eric Domenjoud and Francis Klay.
- Satisfiability in constructor-sharing theories [52]: Cesare Tinelli.
- Unification and matching in hierarchical combinations of equational theories [25,26]: Serdar Erbatur, Deepak Kapur, Andrew Marshall and Paliath Narendran.

Last but not least, a special thanks to Franz Baader for his continuous help. The combination papers (co-)authored by Franz Baader are the most influential in my research activity. This collection of papers (see [11,8,6,7,13,9] for a non-exhaustive list) is a very precious guide for a journey in combination of theories.

References

1. Abadi, M., Cortier, V.: Deciding knowledge in security protocols under equational theories. Theor. Comput. Sci. **367**(1-2), 2–32 (2006)
2. Adi, M.: Calculs Associatifs-Commutatifs — Etude et réalisation du système $UNIF_{AC}$. Ph.D. thesis, Université de Nancy 1 (1991)
3. Adi, M., Kirchner, C.: AC-Unification race: The system solving approach, implementation and benchmarks. J. Symb. Comput. **14**(1), 51–70 (1992)
4. Armando, A., Bonacina, M.P., Ranise, S., Schulz, S.: New results on rewrite-based satisfiability procedures. ACM Trans. Comput. Log. **10**(1), 4:1–4:51 (2009)
5. Armando, A., Ranise, S., Rusinowitch, M.: A rewriting approach to satisfiability procedures. Inf. Comput. **183**(2), 140–164 (2003)
6. Baader, F.: Combination of compatible reduction orderings that are total on ground terms. In: Winskel, G. (ed.) Proceedings of the Twelfth Annual IEEE Symposium on Logic in Computer Science (LICS-97). pp. 2–13. IEEE Computer Society Press, Warsaw, Poland (1997)
7. Baader, F., Schulz, K.: Combination of constraint solvers for free and quasi-free structures. Theoretical Computer Science **192**, 107–161 (1998)
8. Baader, F., Schulz, K.U.: Combination techniques and decision problems for disunification. Theoretical Computer Science **142**(2), 229–255 (1995)
9. Baader, F., Ghilardi, S., Tinelli, C.: A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. Information & Computation **204**(10), 1413–1452 (2006)
10. Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press (1998)
11. Baader, F., Schulz, K.U.: Unification in the union of disjoint equational theories: Combining decision procedures. Journal of Symbolic Computation **21**(2), 211–243 (Feb 1996)
12. Baader, F., Snyder, W.: Unification theory. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning (in 2 volumes), pp. 445–532. Elsevier and MIT Press (2001)

13. Baader, F., Tinelli, C.: Deciding the word problem in the union of equational theories. Information and Computation **178**(2), 346–390 (Dec 2002)
14. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. **4**(3), 217–247 (1994)
15. Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W.: Basic paramodulation. Inf. Comput. **121**(2), 172–192 (1995)
16. Blanchet, B.: Modeling and verifying security protocols with the Applied Pi calculus and proverif. Foundations and Trends in Privacy and Security **1**(1-2), 1–135 (2016)
17. Bonacina, M.P.: A taxonomy of theorem-proving strategies. In: Wooldridge, M.J., Veloso, M. (eds.) Artificial Intelligence Today – Recent Trends and Developments, Lecture Notes in Artificial Intelligence, vol. 1600, pp. 43–84. Springer Berlin Heidelberg (1999)
18. Borovanský, P., Kirchner, C., Kirchner, H., Moreau, P.: ELAN from a rewriting logic point of view. Theor. Comput. Sci. **285**(2), 155–185 (2002)
19. Boudet, A.: Combining unification algorithms. Journal Symbolic Computation **16**(6), 597–626 (1993)
20. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.L. (eds.): All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic, Lecture Notes in Computer Science, vol. 4350. Springer (2007)
21. Comon, H., Haberstrau, M., Jouannaud, J.P.: Syntacticness, cycle-syntacticness, and shallow theories. Inf. Comput. **111**(1), 154–191 (1994)
22. Comon-Lundh, H., Delaune, S.: The finite variant property: How to get rid of some algebraic properties. In: Giesl, J. (ed.) Term Rewriting and Applications, 16th International Conference, RTA 2005, Nara, Japan, April 19-21, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3467, pp. 294–307. Springer (2005)
23. Domenjoud, E., Klay, F., Ringeissen, C.: Combination Techniques for Non-disjoint Equational Theories. In: Bundy, A. (ed.) Proc. of 12th International Conference on Automated Deduction, (CADE'94), Nancy, France. Lecture Notes in Artificial Intelligence, vol. 814, pp. 267–281. Springer-Verlag (1994)
24. Eeralla, A.K., Erbatur, S., Marshall, A.M., Ringeissen, C.: Rule-based unification in combined theories and the finite variant property. In: Martín-Vide, C., Okhotin, A., Shapira, D. (eds.) Language and Automata Theory and Applications - 13th International Conference, LATA 2019, St. Petersburg, Russia, March 26-29, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11417, pp. 356–367. Springer (2019)
25. Erbatur, S., Kapur, D., Marshall, A.M., Narendran, P., Ringeissen, C.: Hierarchical combination. In: Bonacina, M.P. (ed.) Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7898, pp. 249–266. Springer (2013)
26. Erbatur, S., Kapur, D., Marshall, A.M., Narendran, P., Ringeissen, C.: Unification and matching in hierarchical combinations of syntactic theories. In: Lutz, C., Ranise, S. (eds.) Frontiers of Combining Systems - 10th International Symposium, FroCoS 2015, Wroclaw, Poland, September 21-24, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9322, pp. 291–306. Springer (2015)
27. Erbatur, S., Marshall, A.M., Ringeissen, C.: Notions of knowledge in combinations of theories sharing constructors. In: de Moura, L. (ed.) Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg,

Sweden, August 6-11, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10395, pp. 60–76. Springer (2017)
28. Escobar, S., Meadows, C., Meseguer, J.: Maude-NPA: Cryptographic protocol analysis modulo equational properties. In: Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures. Lecture Notes in Computer Science, vol. 5705, pp. 1–50. Springer (2009)
29. Escobar, S., Sasse, R., Meseguer, J.: Folding variant narrowing and optimal variant termination. J. Log. Algebr. Program. **81**(7-8), 898–928 (2012)
30. Jouannaud, J.P., Kirchner, C.: Solving equations in abstract algebras: a rule-based survey of unification. In: Lassez, J.L., Plotkin, G. (eds.) Computational Logic. Essays in honor of Alan Robinson, chap. 8, pp. 257–321. MIT Press, Cambridge (MA, USA) (1991)
31. Jouannaud, J.P., Kirchner, H.: Completion of a set of rules modulo a set of equations. SIAM J. Comput. **15**(4), 1155–1194 (1986)
32. Kapur, D., Narendran, P.: Complexity of unification problems with associative-commutative operators. J. Autom. Reasoning **9**(2), 261–288 (1992)
33. Kapur, D., Narendran, P.: Double-exponential complexity of computing a complete set of AC-unifiers. In: Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS '92), Santa Cruz, California, USA, June 22-25, 1992. pp. 11–21. IEEE Computer Society (1992)
34. Kirchner, C., Klay, F.: Syntactic theories and unification. In: Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990. pp. 270–277. IEEE Computer Society (1990)
35. Kirchner, C., Ringeissen, C.: Rule-based constraint programming. Fundam. Inform. **34**(3), 225–262 (1998)
36. Lynch, C., Morawska, B.: Basic syntactic mutation. In: Voronkov, A. (ed.) Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2392, pp. 471–485. Springer (2002)
37. Meier, S., Schmidt, B., Cremers, C., Basin, D.A.: The TAMARIN prover for the symbolic analysis of security protocols. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8044, pp. 696–701. Springer (2013)
38. Meseguer, J.: Variant-based satisfiability in initial algebras. Sci. Comput. Program. **154**, 3–41 (2018)
39. Moreau, P.E., Ringeissen, C., Vittek, M.: A pattern matching compiler for multiple target languages. In: Hedin, G. (ed.) Compiler Construction, 12th International Conference, CC 2003, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2003, Warsaw, Poland, April 7-11, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2622, pp. 61–76. Springer (2003)
40. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Transations on Programming Languages and Systems **1**(2), 245–257 (Oct 1979)
41. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning (in 2 volumes), pp. 371–443. Elsevier and MIT Press (2001)
42. Nipkow, T.: Combining matching algorithms: The regular case. Journal of Symbolic Computation **12**(6), 633–654 (1991)
43. Nipkow, T.: Proof transformations for equational theories. In: Proceedings of the Fifth Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, Pennsylvania, USA, June 4-7, 1990. pp. 278–288. IEEE Computer Society (1990)

44. Otop, J.: E-unification with constants vs. general E-unification. J. Autom. Reasoning **48**(3), 363–390 (2012)
45. Pigozzi, D.: The joint of equational theories. In: Colloquium Mathematicum. pp. 15–25 (1974)
46. Ringeissen, C.: Combining Decision Algorithms for Matching in the Union of Disjoint Equational Theories. Information and Computation **126**(2), 144–160 (May 1996)
47. Ringeissen, C.: Matching with free function symbols - A simple extension of matching? In: Middeldorp, A. (ed.) Rewriting Techniques and Applications, 12th International Conference, RTA 2001, Utrecht, The Netherlands, May 22-24, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2051, pp. 276–290. Springer (2001)
48. Ringeissen, C.: Matching in a class of combined non-disjoint theories. In: Baader, F. (ed.) Automated Deduction - CADE-19, 19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28 - August 2, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2741, pp. 212–227. Springer (2003)
49. Schmidt-Schauß, M.: Unification in a combination of arbitrary disjoint equational theories. Journal of Symbolic Computation **8**, 51–99 (1989)
50. Schmidt-Schauß, M.: Unification in permutative equational theories is undecidable. J. Symb. Comput. **8**(4), 415–421 (1989)
51. Tidén, E.: Unification in combinations of collapse-free theories with disjoint sets of function symbols. In: Proc. of the 8th International Conference on Automated Deduction, (CADE'86), Oxford, England, July 27 - August 1. Lecture Notes in Computer Science, vol. 230, pp. 431–449. Springer (1986)
52. Tinelli, C., Ringeissen, C.: Unions of Non-Disjoint Theories and Combinations of Satisfiability Procedures. Theoretical Computer Science **290**(1), 291–353 (Jan 2003)
53. Yelick, K.A.: Unification in combinations of collapse-free regular theories. Journal of Symbolic Computation **3**(1/2), 153–181 (1987)